**Как просмотреть содержимое ключей и сертификатов SSL**

# Как просмотреть содержимое ключей и сертификатов SSL

## Просмотр содержимого ключей и сертификатов

Мы можем подробно изучить содержимое всех созданных в OpenSSL файлов, а также при необходимости конвертировать их в другие форматы.

В следующих командах используются тестовые файлы со следующими именами:

* **rootCA.key** - ключ CA
* **rootCA.crt** - сертификат CA
* **mydomain.com.key** - ключ сервера
* **mydomain.com.csr** - запрос за подпись сертификата сайта
* **mydomain.com.crt** - сертификат сайта

Обратите внимание на расширения файлов — они могут отличаться от тех, которые используются в других инструкциях. Например, вместо **.key** и **.crt** может использоваться расширение **.pem**. Расширение файла не имеет особого значения кроме как служить подсказкой пользователю, что именно находится в этом файле. Это же самое касается и имён файлов — вы можете выбирать любые имена.

Все эти файлы являются текстовыми:

|  |  |
| --- | --- |
| 1 | cat rootCA.key |

Там мы увидим примерно следующее:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | -----BEGIN PRIVATE KEY-----  MIIJRAIBADANBgkqhkiG9w0BAQEFAASCCS4wggkqAgEAAoICAQDJBKkr6XzzAcXD  eyDQdvB0SWE2Fl3nqlX/c2RgqMgScXtgidEzOu9ms3Krju5UKLokkQJrZFPMtiIL  MuPJFdYjVyfkfnqlZiouBVgJ60s8NQBBI8KnyyAoJCIFdASoW4Kv5C5LT8pX9eRa  /huJaRJL5XsFUGnTOLvW2ZLN52iAux9CoZlmH6ZF4nuQpblwN0MHULAhze52VNFT  …………………………………………………..  …………………………………………………..  …………………………………………………..  …………………………………………………..  …………………………………………………..  …………………………………………………..  -----END PRIVATE KEY----- |

[](https://hackware.ru/wp-content/uploads/2020/06/rsa-key.png)

Если вам эти строки кажутся знакомыми на кодировку **Base64**, то вы совершенно правы — это она и есть. (Смотрите также «[Как быстро узнать и преобразовать кодировку](https://hackware.ru/?p=9551)»).

Этот формат, называемый форматом PEM, расшифровывается как Privacy Enhanced Mail.

PEM — это текстовое представление реального двоичного ключа или сертификата в формате DER. Представляет собой двоичного формата DER в кодировке base64 и с дополнительными строками «-----BEGIN PRIVATE KEY-----», «-----BEGIN CERTIFICATE-----» и другими в начале файла и строками «-----END PRIVATE KEY-----», «-----END CERTIFICATE-----» в конце файла.

Мы можем хранить двоичную версию файла только с кодировкой DER, но наиболее распространенным способом является версия PEM.

Вы можете увидеть структуру приватного следующей командой:

|  |  |
| --- | --- |
| 1 | openssl rsa -text -in rootCA.key |

Опция -in ИМЯ\_ФАЙЛА указывает имя файла ввода для чтения ключа или стандартный ввод, если эта опция не указана. Если ключ зашифрован, будет запрошен пароль.

Опция -text печатает различные компоненты открытого или закрытого ключа в виде простого текста в дополнение к закодированной версии.

Пример вывода:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | RSA Private-Key: (4096 bit, 2 primes)  modulus:      00:c9:04:a9:2b:e9:7c:f3:01:c5:c3:7b:20:d0:76:      [...]  publicExponent: 65537 (0x10001)  privateExponent:      1f:86:71:99:87:66:a7:1d:b2:0c:34:35:33:3c:53:      [...]  prime1:      00:f0:af:82:a6:f1:40:85:ee:c0:77:cc:41:ce:11:      [...]  prime2:      00:d5:cf:03:c6:2a:01:79:9a:e3:1d:ec:1b:52:40:      [...]  exponent1:      00:d7:7e:ed:65:f7:9f:a3:cb:2e:bc:94:3f:5e:f8:      [...]  exponent2:      00:ae:a1:5e:db:c4:03:60:67:79:89:3f:07:31:ae:      [...]  coefficient:      00:e4:7d:de:4e:00:a0:8d:c4:5a:14:93:b6:7f:c9:      [...]  writing RSA key  -----BEGIN RSA PRIVATE KEY-----  [...]  -----END RSA PRIVATE KEY----- |

Любой формат ключа на самом деле является контейнером для набора длинных чисел. Все остальные данные можно считать «шумом».

Закрытый ключ содержит: модуль (modulus), частный показатель (privateExponent), открытый показатель (publicExponent), простое число 1 (prime1), простое число 2 (prime2), показатель степени 1 (exponent1), показатель степени 2 (exponent2) и коэффициент (coefficient).

Открытый ключ содержит только модуль (modulus) и открытый показатель (publicExponent).

Вы можете из влечь из файла ключей публичный ключ:

|  |  |
| --- | --- |
| 1 | openssl rsa -in rootCA.key -pubout -out rootCA-public.key |

По умолчанию выводится закрытый ключ: с опцией **-pubout**вместо него будет выведен открытый ключ. Эта опция устанавливается автоматически, если ввод является открытым ключом.

Следующая команда покажет информацию о публичном ключе:

|  |  |
| --- | --- |
| 1 | openssl rsa -text -in rootCA-public.key -pubin |

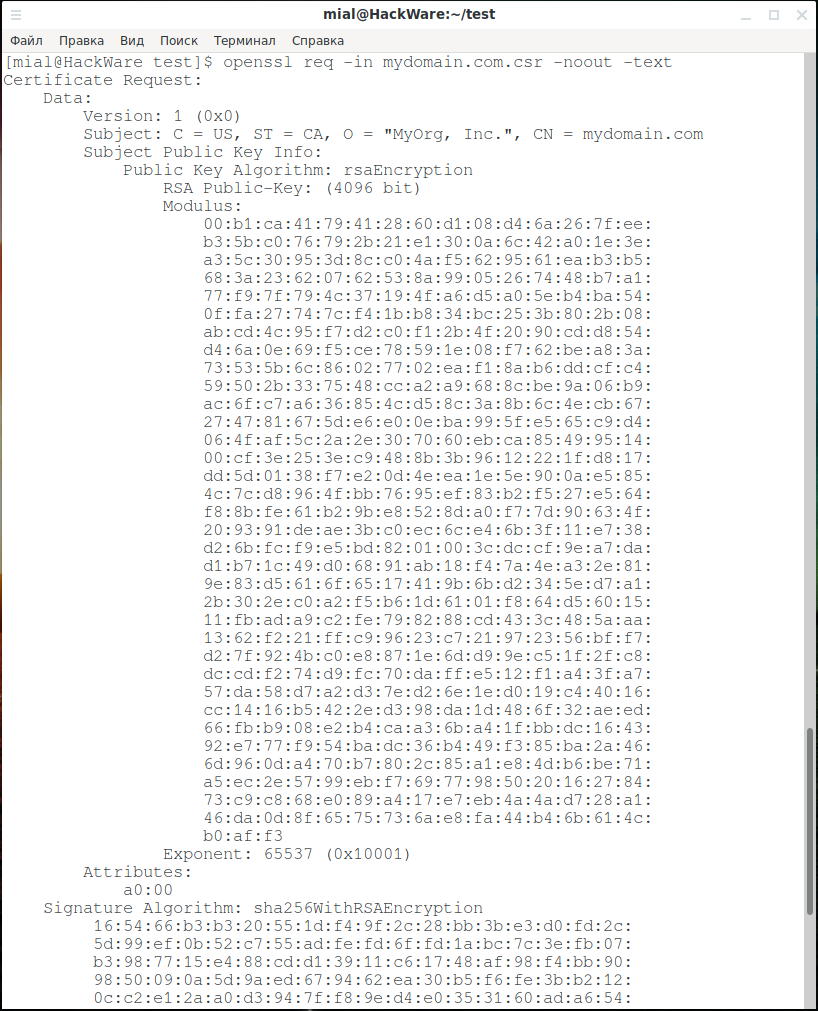
По умолчанию из входного файла считывается закрытый ключ. Используемая в предыдущей команде опция **-pubin** делает так, что вместо приватного ключа читается открытый ключ.

Можно также добавить опцию **-noout** — с ней будет выведена та же самая информация, но не будет показана кодированная версия ключа.

|  |  |
| --- | --- |
| 1 | openssl rsa -text -in rootCA-public.key -pubin -noout |

С такими же опциями, но уже используя команду **req**, можно изучить содержимое запроса на подпись сертификата:

|  |  |
| --- | --- |
| 1 | openssl req -in mydomain.com.csr -noout -text |

[](https://hackware.ru/wp-content/uploads/2020/06/openssl-req.png)

При создании SSL сертификата мы создали две пары ключей (корневые и для домена), то есть это файлы **rootCA.key** и **mydomain.com.key**, но по своей технической сути они идентичны.

Что касается сертификатов, которых у нас тоже два (**rootCA.crt** и **mydomain.com.crt**), то по своей природе они не являются одинаковыми: корневой сертификат является самоподписанным, а сертификат домена подписан приватным корневым ключом.

Информацию о содержимом сертификатов можно посмотреть командой **x509** (остальные опции нам уже знакомы):

|  |  |
| --- | --- |
| 1  2 | openssl x509 -in rootCA.crt -noout -text  openssl x509 -in mydomain.com.crt -text -noout |

Самоподписанные сертификаты обычно содержат только самые основные данные сертификатов, как показано в предыдущем примере. Для сравнения, сертификаты, выданные общедоступными центрами сертификации, гораздо интереснее, поскольку они содержат ряд дополнительных полей (с помощью механизма расширений X.509).

Сертификат (цепочку сертификатов) любого сайта вы можете получить следующей командой (замените **w-e-b.site** на интересующий вас сайт):

|  |  |
| --- | --- |
| 1 | openssl s\_client -showcerts -connect w-e-b.site:443 </dev/null |

[](https://hackware.ru/wp-content/uploads/2020/06/w-e-b.site_.cert_.png)

Вы увидите сертификаты (один или несколько), найдите по полю **CN** тот, который вас интересует, например, сертификат домена **w-e-b.site**:

|  |  |
| --- | --- |
| 1 | 0 s:CN = w-e-b.site |

И скопируйте содержимое начиная с **-----BEGIN CERTIFICATE-----** и заканчивая **-----END CERTIFICATE-----**. Затем сохраните это в файл.

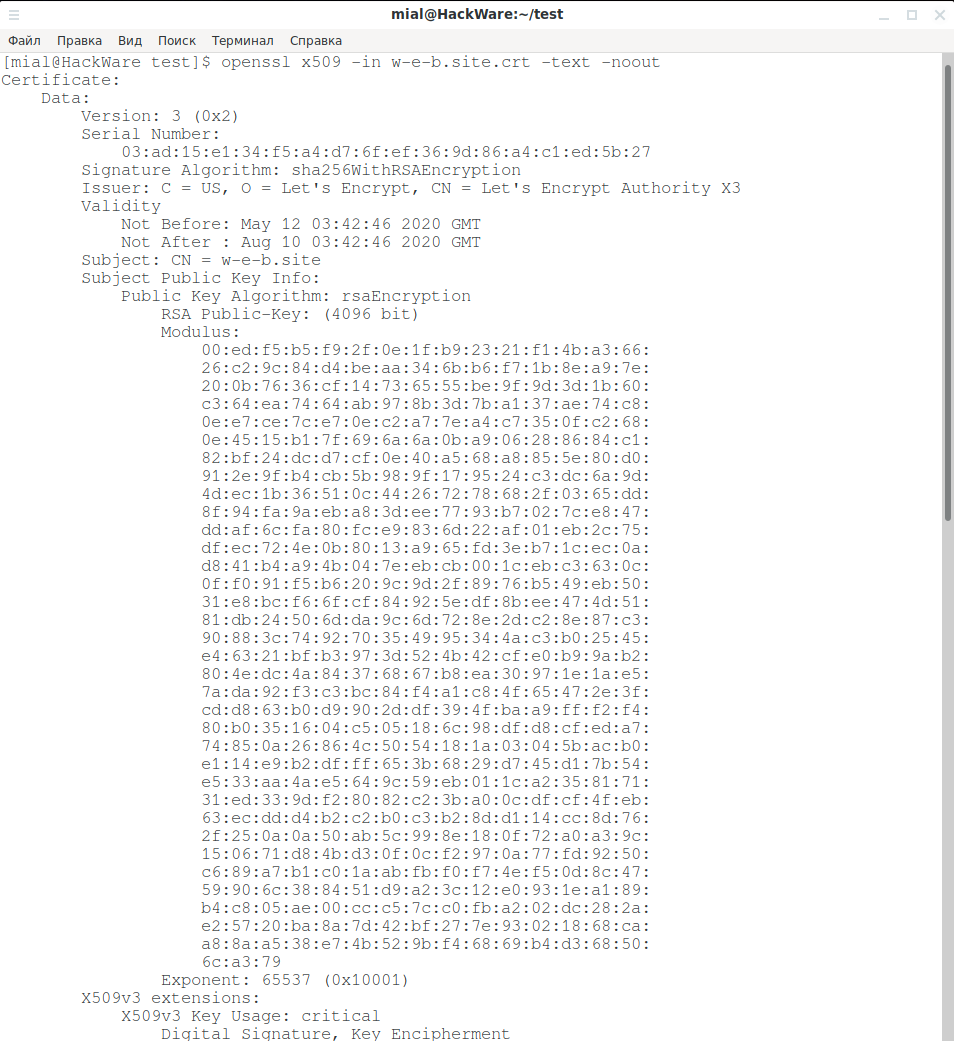
Вы также можете сохранить сертификат центра сертификации и изучить его:

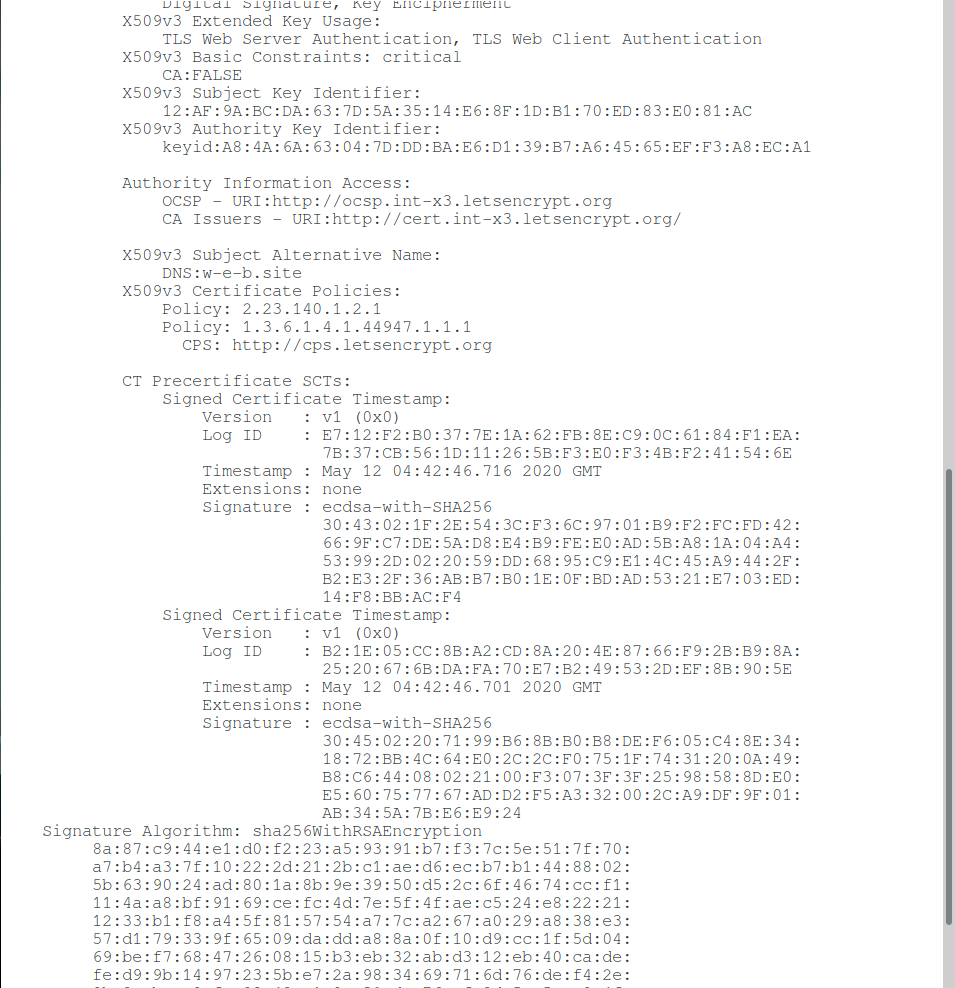
[](https://hackware.ru/wp-content/uploads/2020/06/x3-cert.png)

К примеру, сертификат сайта w-e-b.site я сохранил в файл **w-e-b.site.crt**, для просмотра информации о нём:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -text -noout |

Пример вывода:

[](https://hackware.ru/wp-content/uploads/2020/06/openssl-x509.png)

[](https://hackware.ru/wp-content/uploads/2020/06/openssl-x509-2.png)

**Issuer** указывает на организацию, которая выдала (подписала) сертификат:

|  |  |
| --- | --- |
| 1 | Issuer: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3 |

Validity — срок действия сертификата:

|  |  |
| --- | --- |
| 1  2  3 | Validity      Not Before: May 12 03:42:46 2020 GMT      Not After : Aug 10 03:42:46 2020 GMT |

Subject: CN — домен (IP адрес) для которого предназначен сертификат:

|  |  |
| --- | --- |
| 1 | Subject: CN = w-e-b.site |

Поддомены в группе X509v3 extensions → X509v3 Subject Alternative Name (подробности чуть позже):

|  |  |
| --- | --- |
| 1  2 | X509v3 Subject Alternative Name:      DNS:w-e-b.site, DNS:www.w-e-b.site |

Теперь бегло рассмотрим расширения X.509.

Расширение **Basic Constraints** используется для маркировки сертификатов как принадлежащих ЦС, давая им возможность подписывать другие сертификаты. В сертификатах, отличных от CA, это расширение будет либо пропущено, либо будет установлено значение **CA**, равное **FALSE**. Это расширение является критическим, что означает, что все программные сертификаты должны понимать его значение.

|  |  |
| --- | --- |
| 1  2 | X509v3 Basic Constraints: critical      CA:FALSE |

Расширения **Key Usage (KU)** и **Extended Key Usage (EKU)** ограничивают возможности использования сертификата. Если эти расширения присутствуют, то разрешены только перечисленные варианты использования. Если расширения отсутствуют, ограничений на использование нет. То, что вы видите в этом примере, типично для сертификата веб-сервера, который, например, не позволяет подписывать код:

|  |  |
| --- | --- |
| 1  2  3  4 | X509v3 Key Usage: critical      Digital Signature, Key Encipherment  X509v3 Extended Key Usage:      TLS Web Server Authentication, TLS Web Client Authentication |

Расширение **CRL Distribution Points** перечисляет адреса, по которым можно найти информацию о списке отзыва сертификатов (CRL) ЦС. Эта информация важна в случаях, когда сертификаты необходимо отозвать. CRL — это подписанные CA списки отозванных сертификатов, публикуемые через регулярные промежутки времени (например, семь дней).

|  |  |
| --- | --- |
| 1  2  3 | X509v3 CRL Distribution Points:      Full Name:        URI:http://crl.starfieldtech.com/sfs3-20.crl |

Примечание: возможно, вы заметили, что местоположение CRL не использует защищённый сервер, и вам может быть интересно, является ли ссылка небезопасной. Не является. Поскольку каждый CRL подписан центром сертификации, который его выпустил, браузеры могут проверить его целостность. В том же случае, если бы CRL были доступны по TLS (адрес включал бы в себя протокол HTTPS), то браузеры могут столкнуться с проблемой «курицы и яйца», в которой они хотят проверить статус отзыва сертификата, используемого сервером, доставляющим сам CRL!

Расширение **Certificate Policies** используется для указания политики, в соответствии с которой был выпущен сертификат. Например, именно здесь можно найти индикаторы расширенной проверки (EV). Индикаторы представлены в форме уникальных идентификаторов объектов (OID) и являются уникальными для выдающего ЦС. Кроме того, это расширение часто содержит один или несколько пунктов CPS, которые обычно являются веб-страницами или документами PDF.

|  |  |
| --- | --- |
| 1  2  3  4 | X509v3 Certificate Policies:      Policy: 2.23.140.1.2.1      Policy: 1.3.6.1.4.1.44947.1.1.1        CPS: http://cps.letsencrypt.org |

Расширение **Authority Information Access (AIA)** обычно содержит две важные части информации. Во-первых, он перечисляет адрес ответчика CA OCSP, который можно использовать для проверки отзыва сертификатов в режиме реального времени. Расширение также может содержать ссылку, где находится сертификат эмитента (следующий сертификат в цепочке). В наши дни серверные сертификаты редко подписываются непосредственно доверенными корневыми сертификатами, а это означает, что пользователи должны включать в свою конфигурацию один или несколько промежуточных сертификатов. Ошибки легко сделать, и сертификаты будут признаны недействительными. Некоторые клиенты (например, Internet Explorer) будут использовать информацию, представленную в этом расширении для исправления неполной цепочки сертификатов, но многие клиенты этого не сделают.

|  |  |
| --- | --- |
| 1  2  3 | Authority Information Access:      OCSP - URI:http://ocsp.int-x3.letsencrypt.org      CA Issuers - URI:http://cert.int-x3.letsencrypt.org |

Расширения **Subject Key Identifier** и **Authority Key Identifier** устанавливают уникальные идентификаторы ключа субъекта и ключа авторизации соответственно. Значение, указанное в расширении **Authority Key Identifier** сертификата, должно соответствовать значению, указанному в расширении **Subject Key Identifier** в выдающем сертификате. Эта информация очень полезна в процессе построения пути сертификации, когда клиент пытается найти все возможные пути от конечного (серверного) сертификата до доверенного корня. Центры сертификации часто используют один закрытый ключ с несколькими сертификатами, и это поле позволяет программному обеспечению надёжно определять, какой сертификат может быть сопоставлен с каким ключом. В реальном мире многие цепочки сертификатов, предоставляемые серверами, недействительны, но этот факт часто остаётся незамеченным, поскольку браузеры могут находить альтернативные пути доверия.

|  |  |
| --- | --- |
| 1  2  3  4 | X509v3 Subject Key Identifier:      12:AF:9A:BC:DA:63:7D:5A:35:14:E6:8F:1D:B1:70:ED:83:E0:81:AC  X509v3 Authority Key Identifier:      keyid:A8:4A:6A:63:04:7D:DD:BA:E6:D1:39:B7:A6:45:65:EF:F3:A8:EC:A1 |

Наконец, расширение **Subject Alternative Name** используется для перечисления всех имен хостов, для которых действителен сертификат. Это расширение раньше было необязательным; если его нет, клиенты возвращаются к использованию информации, представленной в **Common Name (CN)**, которое является частью поля «**Subject**». Если расширение присутствует, то содержимое поля **CN** игнорируется во время проверки.

|  |  |
| --- | --- |
| 1  2 | X509v3 Subject Alternative Name:      DNS:w-e-b.site, DNS:www.w-e-b.site |

Рассмотрим опции команды **x509**, которые позволяют извлечь разнообразную информацию из сертификатов.

Чтобы показать издателя сертификата используйте опцию **-issuer**:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -issuer |

Пример вывода:

|  |  |
| --- | --- |
| 1 | issuer=C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3 |

Опция **-fingerprint** вычисляет и выводит дайджест DER-кодированной версии всего сертификата. Это обычно называют «отпечатком». Из-за характера дайджестов сообщений, отпечаток сертификата является уникальным для этого сертификата, и два сертификата с одинаковым отпечатком могут считаться одинаковыми. Для сертификатов обычно не нужно сверять сертификаты по отпечаткам, но это имеет смысл при использовании самоподписанных сертификатов (например, получении [сертификата для VNC сессии](https://hackware.ru/?p=12588#2119), когда нет другого способа проверить, что сертификат не был подменён при пересылке). В этом случае можно сверить отпечаток сертификата, например, по телефону или электронной почте.

Для показа отпечатка сертификата:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -fingerprint |

Пример вывода:

|  |  |
| --- | --- |
| 1 | SHA1 Fingerprint=43:4E:55:5B:27:09:33:00:3F:43:B0:B7:B2:5E:96:D5:10:42:3B:44 |

Чтобы вывести сертификат в виде строки [символов в стиле C — char](https://hackware.ru/?p=12978#61), используйте опцию **-C**:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -C |

Чтобы вывести расширения сертификата в текстовой форме, используйте опцию **-ext**. Несколько расширений можно перечислить через запятую, например "**subjectAltName,subjectKeyIdentifier**". Чтобы посмотреть весь список расширений:

|  |  |
| --- | --- |
| 1 | man x509v3\_config |

Пример команды для вывода альтернативных имён в домене:

|  |  |
| --- | --- |
| 1 | openssl x509 -in suip.biz.cert -noout -ext subjectAltName |

Пример информации об альтернативных именах домена:

|  |  |
| --- | --- |
| 1  2 | X509v3 Subject Alternative Name:      DNS:suip.biz, DNS:www.suip.biz |

Для вывода почтовых адресов, содержащихся в сертификате, используйте опцию **-email**. Адреса электронной почты могут отсутствовать в сертификате.

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -email |

Для вывода адресов респондентов OCSP, если они присутствуют в сертификате, укажите опцию **-ocsp\_uri**:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -ocsp\_uri |

Пример вывода:

|  |  |
| --- | --- |
| 1 | http://ocsp.int-x3.letsencrypt.org |

Для показа дат из сертификата имеются следующие опции:

* **-startdate**: Распечатывает дату начала сертификата, то есть дату notBefore (не ранее).
* **-enddate**: Распечатывает дату истечения срока действия сертификата, то есть дату notAfter (не позднее).
* **-dates**: Распечатывает даты начала и окончания срока действия сертификата.

Для вывода имени subject укажите опцию **-subject**:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -subject |

Пример вывода:

|  |  |
| --- | --- |
| 1 | subject=CN = w-e-b.site |

Чтобы показать имя subject сертификата в форме RFC2253 используйте сочетание опций **-subject -nameopt RFC2253**:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -subject -nameopt RFC2253 |

Пример вывода:

|  |  |
| --- | --- |
| 1 | subject=CN=w-e-b.site |

Пример вывода имени subject сертификата в форме схемы на терминале, поддерживающем UTF8:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -subject -nameopt oneline,-esc\_msb |

Опция **-serial** выводит серийный номер:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -serial |

Пример вывода:

|  |  |
| --- | --- |
| 1 | serial=03AD15E134F5A4D76FEF369D86A4C1ED5B27 |

Чтобы извлечь публичный ключ из сертификата используйте опцию **-pubkey**:

|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -pubkey |

**Для глубокого понимания OpenSSL смотрите также полное руководство: «[OpenSSL: принципы работы, создание сертификатов, аудит](https://hackware.ru/?p=12982" \t "_blank)».**

## Связанные статьи:

* [Как создать сертификаты SSL (TLS)](https://zalinux.ru/?p=4174) (44.9%)
* [Как верифицировать сертификат SSL](https://zalinux.ru/?p=4181) (44.9%)
* [Как включить SMPTS (465) postfix в Linux](https://zalinux.ru/?p=431) (37.7%)
* [Исправление смешенного контента HTTPS в системе пользовательского поиска Google](https://zalinux.ru/?p=616) (37.7%)
* [Установка последних версий Node.js и NPM в Linux](https://zalinux.ru/?p=1109) (37.7%)
* [Практические примеры использования команды find в Linux](https://zalinux.ru/?p=2745) (RANDOM - 12.4%)

# Как создать сертификаты SSL (TLS)

## Создание корневого приватного ключа

Внимание: этот ключ используется для подписи запросов сертификатов, любой, кто получил этот ключ, может подписывать сертификаты от вашего имени, поэтому храните его в безопасном месте:

Генерация приватного ключа RSA используя параметры по умолчанию (ключ будет сохранён в файл с именем **rootCA.key**):

|  |  |
| --- | --- |
| 1 | openssl genpkey -algorithm RSA -out rootCA.key |

Опция **-out** указывает на имя файла для сохранения, без этой опции файл будет выведен в стандартный вывод (на экран). Имя выходного файла не должно совпадать с именем входного файла.

Для безопасности ключа его следует защитить паролем. Генерация приватного ключа RSA используя 128-битное AES шифрование ***(-aes-128-cbc)*** и парольную фразу "hello" ***(-pass pass:hello)***:

|  |  |
| --- | --- |
| 1 | openssl genpkey -algorithm RSA -out rootCA.key -aes-128-cbc -pass pass:hello |

Конечно, опцию **-pass pass:hello** можно не указывать, тогда вам будет предложено ввести пароль во время генерации ключа.

Список поддерживаемых симметричных алгоритмов шифрования приватного ключа можно узнать в документации (раздел **SUPPORTED CIPHERS**):

|  |  |
| --- | --- |
| 1 | man enc |

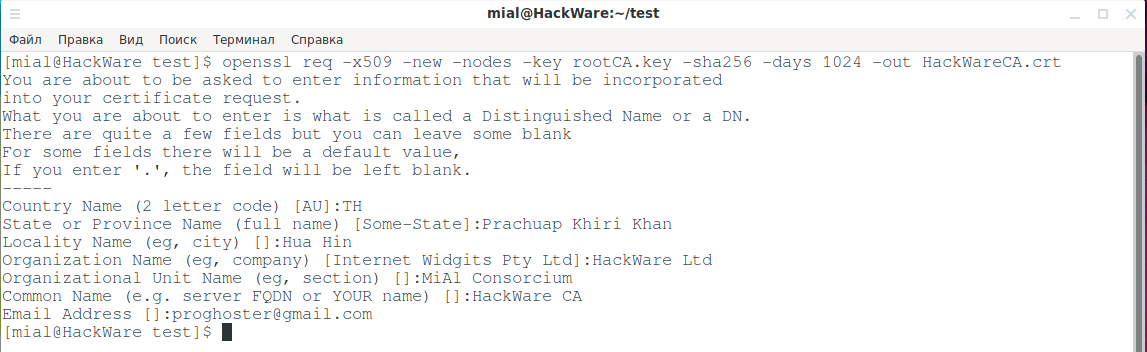
Если для генерируемого ключа не указано количество бит, то по умолчанию используется 2048, вы можете указать другое количество бит с помощью команды вида (будет создан 4096-битный ключ):

|  |  |
| --- | --- |
| 1 | openssl genpkey -algorithm RSA -out rootCA.key -aes-128-cbc -pkeyopt rsa\_keygen\_bits:4096 |

## Создание самоподписанного корневого сертификата

|  |  |
| --- | --- |
| 1 | openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.crt |

Здесь мы использовали наш корневой ключ для создания корневого сертификата (файл **rootCA.crt**), который должен распространяться на всех компьютерах, которые нам доверяют. А приватный ключ (файл **rootCA.key**) должен быть секретным, поскольку он будет использоваться для подписи сертификатов серверов.

[](https://hackware.ru/wp-content/uploads/2020/06/HackWareCA.crt_.png)

Создание сертификатов (делается для каждого домена) включает в себя несколько этапов. Эту процедуру необходимо выполнить для каждого домена/сервера, которым требуется доверенный сертификат от нашего ЦС.

## Создание приватного ключа сертификата

|  |  |
| --- | --- |
| 1 | openssl genpkey -algorithm RSA -out mydomain.com.key |

Обратите внимание, что это та же самая команда, которой мы создавали пару приватный-публичный ключ Центра Сертификации (изменено только имя файла с ключами).

## Создание файла с запросом на подпись сертификата (csr)

Получив закрытый ключ, вы можете приступить к созданию запроса на подпись сертификата — Certificate Signing Request (CSR). Это официальный запрос к CA о подписании сертификата, который содержит открытый ключ объекта, запрашивающего сертификат, и некоторую информацию об объекте. Все эти данные будут частью сертификата. CSR всегда подписывается закрытым ключом, соответствующим открытому ключу, который он несёт.

Создание CSR обычно представляет собой интерактивный процесс, в ходе которого вы будете предоставлять элементы отличительного имени сертификата (вводить информацию о стране, городе, организации, email и т.д.). Внимательно прочитайте инструкции, предоставленные инструментом openssl; если вы хотите, чтобы поле было пустым, вы должны ввести одну точку (**.**) в строке, а не просто нажать «Enter». Если вы сделаете последнее, OpenSSL заполнит соответствующее поле CSR значением по умолчанию. (Такое поведение не имеет никакого смысла при использовании с конфигурацией OpenSSL по умолчанию, что и делают практически все. Это имеет смысл, когда вы осознаете, что можете изменить значения по умолчанию, либо изменив конфигурацию OpenSSL, либо предоставив свои собственные конфигурации в файлах).

В запросе на подпись сертификата вы указываете информацию для сертификата, который хотите сгенерировать. Этот запрос будет обработан владельцем корневого ключа (в данном случае вы его создали ранее) для генерации сертификата.

Важно: имейте в виду, что при создании запроса на подпись важно указать **Common Name**, предоставляющее IP-адрес или доменное имя для службы, в противном случае сертификат не может быть проверен.

Я опишу здесь два способа:

* Метод А (интерактивный)

Если вы создадите CSR таким способом, openssl задаст вам вопросы о сертификате, который необходимо сгенерировать, например, сведения об организации и **Common Name (CN)**, которое является веб-адресом, для которого вы создаёте сертификат, например, mydomain.com.

|  |  |
| --- | --- |
| 1 | openssl req -new -key mydomain.com.key -out mydomain.com.csr |

Метод Б (в одну команду без запросов)

Этот метод генерирует тот же результат, что и метод A, но он подходит для использования в вашей автоматизации.

|  |  |
| --- | --- |
| 1 | openssl req -new -sha256 -key mydomain.com.key -subj "/C=US/ST=CA/O=MyOrg, Inc./CN=mydomain.com" -out mydomain.com.csr |

Если вам нужно передать дополнительную конфигурацию, вы можете использовать параметр **-config**, например, здесь я хочу добавить альтернативные имена в мой сертификат.

|  |  |
| --- | --- |
| 1 | openssl req -new -sha256 -key mydomain.com.key -subj "/C=US/ST=CA/O=MyOrg, Inc./CN=mydomain.com" -reqexts SAN -config <(cat /etc/ssl/openssl.cnf <(printf "\n[SAN]\nsubjectAltName=DNS:mydomain.com,DNS:www.mydomain.com")) -out mydomain.com.csr |

## Проверка содержимого CSR

После создания CSR используйте его, чтобы подписать собственный сертификат и/или отправить его в общедоступный центр сертификации и попросить его подписать сертификат. Оба подхода описаны в следующих разделах. Но прежде чем сделать это, неплохо бы ещё раз проверить правильность CSR. Это делается так:

|  |  |
| --- | --- |
| 1 | openssl req -in mydomain.com.csr -noout -text |

## Создание сертификата

Создайте сертификат, используя csr для mydomain.com, корневые ключ и сертификат CA.

Если вы устанавливаете сервер TLS для своего собственного использования, вы, вероятно, не хотите идти в ЦС для покупки публично доверенного сертификата. Намного проще использовать сертификат, подписанный вашим собственным CA. Если вы являетесь пользователем Firefox, при первом посещении веб-сайта вы можете создать исключение для сертификата, после которого сайт будет защищён так, как если бы он был защищён общедоступным сертификатом.

Если у вас уже есть CSR, создайте сертификат, используя следующую команду:

|  |  |
| --- | --- |
| 1 | openssl x509 -req -in mydomain.com.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out mydomain.com.crt -days 500 -sha256 |

В результате выполнения этих команд были созданы следующие файлы:

* **rootCA.key** — приватный ключ Центра Сертификации, должен храниться в секрете в CA
* **rootCA.crt** — публичный корневой сертификат Центра Сертификации — должен быть установлен на всех пользовательских устройствах
* **mydomain.com.key** — приватный ключ веб-сайта, должен храниться в секрете на сервере. Указывает в конфигурации виртуального хоста при настройке веб-сервера
* **mydomain.com.csr** — запрос на подпись сертификата, после создания сертификата этот файл не нужен, его можно удалить
* **mydomain.com.crt** — сертификат сайта. Указывает в конфигурации виртуального хоста при настройке веб-сервера, не является секретным.

**Для глубокого понимания OpenSSL смотрите также полное руководство: «[OpenSSL: принципы работы, создание сертификатов, аудит](https://hackware.ru/?p=12982" \t "_blank)».**

## Связанные статьи:

* [Как добавить сертификат Центра Сертификации (CA) в доверенные в Linux](https://zalinux.ru/?p=4166) (60%)
* [Где в Linux хранятся корневые сертификаты Центров Сертификации (CA)](https://zalinux.ru/?p=4170) (60%)
* [Как верифицировать сертификат SSL](https://zalinux.ru/?p=4181) (49.9%)
* [Установка и использование Docker в Linux (Debian, Ubuntu, Arch Linux)](https://zalinux.ru/?p=1024) (42%)
* [Как просмотреть содержимое ключей и сертификатов SSL](https://zalinux.ru/?p=4178) (35.2%)
* [Как записать символы строки в обратном порядке](https://zalinux.ru/?p=3830) (RANDOM - 16.9%)

# Как верифицировать сертификат SSL

Для проверки действительности сертификата используется команда

|  |  |
| --- | --- |
| 1 | openssl verify СЕРТИФИКАТ |

В разделе «[Как создать сертификаты SSL (TLS) для сайтов](https://hackware.ru/?p=12982#4)» мы создали сертификат **mydomain.com.crt**, попробуем проверить его:

|  |  |
| --- | --- |
| 1 | openssl verify mydomain.com.crt |

Вывод:

|  |  |
| --- | --- |
| 1  2  3 | C = US, ST = CA, O = "MyOrg, Inc.", CN = mydomain.com  error 20 at 0 depth lookup: unable to get local issuer certificate  error mydomain.com.crt: verification failed |

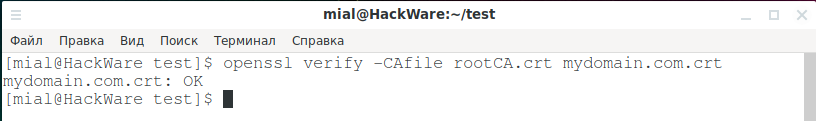
Верификация провалилась — причина в том, что мы использовали для подписи сертификата приватный ключ от своего собственного CA, который создали чуть ранее.

С помощью опции **-CAfile** можно указать сертификат доверенного Центра Сертификации:

|  |  |
| --- | --- |
| 1 | openssl verify -CAfile rootCA.crt mydomain.com.crt |

Как можно увидеть, проверка подлинности прошла успешно:

|  |  |
| --- | --- |
| 1 | mydomain.com.crt: OK |

[](https://hackware.ru/wp-content/uploads/2020/06/openssl-verify.png)

Чтобы было поинтереснее, возьмём реальные сертификаты сайта и промежуточного центра сертификации:

|  |  |
| --- | --- |
| 1 | openssl s\_client -showcerts -connect w-e-b.site:443 </dev/null |

[](https://hackware.ru/wp-content/uploads/2020/06/w-e-b.site_.cert_.png)

Вы увидите сертификаты (один или несколько), найдите по полю **CN** сначала сертификат домена **w-e-b.site**:

|  |  |
| --- | --- |
| 1 | 0 s:CN = w-e-b.site |

И скопируйте содержимое начиная с **-----BEGIN CERTIFICATE-----** и заканчивая **-----END CERTIFICATE-----**. Затем сохраните это в файл **w-e-b.site.crt**.

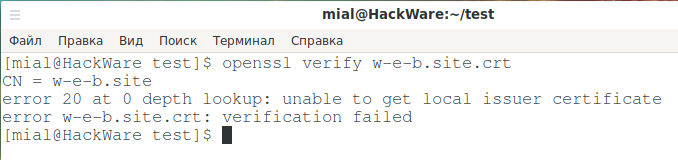
Затем найдите сертификат центра сертификации и сохраните его в файл **x3.crt**:

[](https://hackware.ru/wp-content/uploads/2020/06/x3-cert.png)

Итак, для наших экспериментов у нас появилось два новых файла: **w-e-b.site.crt** (сертификат сайта) и **x3.crt** (сертификат промежуточного Центра Сертификации).

Попробуем проверить действительность сертификата сайта:

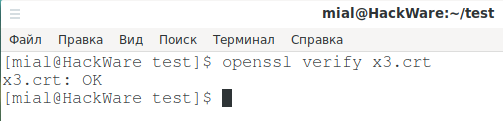
|  |  |
| --- | --- |
| 1 | openssl verify w-e-b.site.crt |

[](https://hackware.ru/wp-content/uploads/2020/06/verification-failed.png)

Поскольку сайт открывается нормально и веб браузер показывает, что с сертификатом сайта всё в порядке, то тот факт, что проверка подлинности не пройдена может показаться странным.

Проверим подлинность промежуточного сертификата организации, которая подписала сертификат сайта:

|  |  |
| --- | --- |
| 1 | openssl verify x3.crt |

[](https://hackware.ru/wp-content/uploads/2020/06/verify.png)

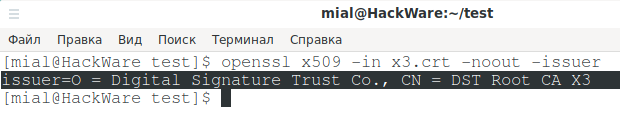
Как можно увидеть на скриншоте, с этим сертификатом всё в порядке.

Дело в том, что команда **openssl verify** сверяет подпись в сертификате с коревыми Центрами Сертификации, чьи сертификаты установлены в операционной системе как доверенные. Мы можем убедиться в этом. Следующая команда покажет организацию, подписавшую сертификат **x3.crt**:

|  |  |
| --- | --- |
| 1 | openssl x509 -in x3.crt -noout -issuer |

Вывод:

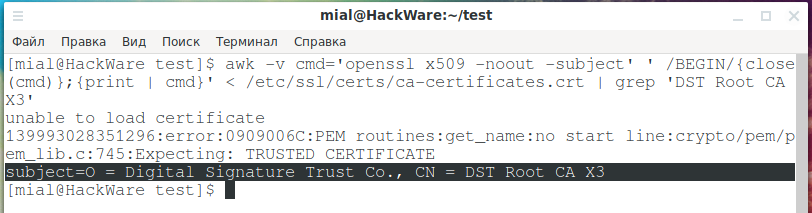
|  |  |
| --- | --- |
| 1 | issuer=O = Digital Signature Trust Co., CN = DST Root CA X3 |

[](https://hackware.ru/wp-content/uploads/2020/06/issuer.png)

Имя издателя в поле **CN** (Common Name) **DST Root CA X3**.

Убедимся, что сертификат этого CA действительно есть в системе:

|  |  |
| --- | --- |
| 1 | awk -v cmd='openssl x509 -noout -subject' ' /BEGIN/{close(cmd)};{print | cmd}' < /etc/ssl/certs/ca-certificates.crt | grep 'DST Root CA X3' |

[](https://hackware.ru/wp-content/uploads/2020/06/dst-root-ca-x3.png)

Если мы посмотрим, кто выдал сертификат сайта [w-e-b.site](https://w-e-b.site/):

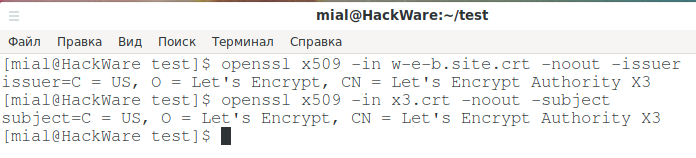
|  |  |
| --- | --- |
| 1 | openssl x509 -in w-e-b.site.crt -noout -issuer |

То увидим, что это **Let's Encrypt Authority X3**.

Посмотрим имя организации в сертификате **x3.crt**:

|  |  |
| --- | --- |
| 1 | openssl x509 -in x3.crt -noout -subject |

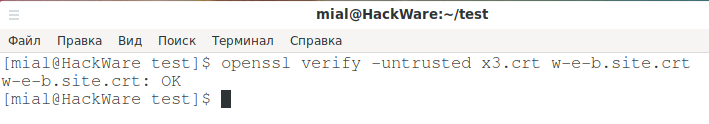
Имя совпало — это также **Let's Encrypt Authority X3**.

[](https://hackware.ru/wp-content/uploads/2020/06/cert-issuer-subject.png)

То есть в целом цепочка понятна — сертификат **w-e-b.site.crt** выдан организацией **Let's Encrypt Authority X3**, а её полномочия по выдаче сертификатов заверяет **DST Root CA X3**, которая в операционной системе находится среди доверенных корневых Центрах Сертификации.

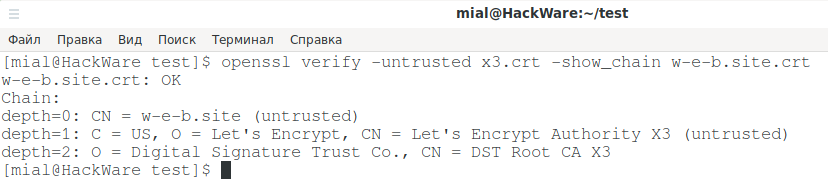
Чтобы автоматически выполнить проверку всей цепочки с помощью **openssl verify**, нам нужно указать промежуточный сертификат после опции **-untrusted**. Если промежуточных сертификатов несколько, то нужно указать их все — опцию **-untrusted** можно использовать несколько раз.

|  |  |
| --- | --- |
| 1 | openssl verify -untrusted x3.crt w-e-b.site.crt |

[](https://hackware.ru/wp-content/uploads/2020/06/crt-ok.png)

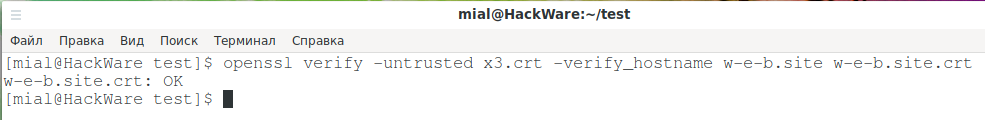
Опция **-show\_chain** покажет информацию о цепочке сертификатов, которая была построена:

|  |  |
| --- | --- |
| 1 | openssl verify -untrusted x3.crt -show\_chain w-e-b.site.crt |

[](https://hackware.ru/wp-content/uploads/2020/06/show_chain.png)

Опции **-verify\_hostname ИМЯ-ХОСТА** и **-verify\_ip IP** позволяют проверить, действительно ли сертификат выдан для указанного хоста или IP адреса. Чтобы проверка прошла успешно, имя хоста должно быть в Subject Alternative Name или Common Name в subject сертификата. А IP адрес должен быть в Subject Alternative Name в subject сертификата. Пример:

|  |  |
| --- | --- |
| 1 | openssl verify -untrusted x3.crt -verify\_hostname w-e-b.site w-e-b.site.crt |

[](https://hackware.ru/wp-content/uploads/2020/06/verify_hostname.png)

Следующими тремя командами вы можете проверить, соответствует ли SSL сертификат приватному ключу:

Для SSL сертификата выполните команду вида:

|  |  |
| --- | --- |
| 1 | openssl x509 -noout -modulus -in ФАЙЛ.crt | openssl md5 |

Для приватного ключа RSA выполните команду вида:

|  |  |
| --- | --- |
| 1 | openssl rsa -noout -modulus -in ФАЙЛ.key | openssl md5 |

Для CSR:

|  |  |
| --- | --- |
| 1 | openssl req -noout -modulus -in ФАЙЛ.csr | openssl md5 |

Замените **ФАЙЛ** на имя ваших файлов. Для всего перечисленного выше — SSL сертификата, приватного ключа и запроса на подпись — хеш должен быть одинаковым.

**Для глубокого понимания OpenSSL смотрите также полное руководство: «[OpenSSL: принципы работы, создание сертификатов, аудит](https://hackware.ru/?p=12982" \t "_blank)».**

## Связанные статьи:

* [Как создать сертификаты SSL (TLS)](https://zalinux.ru/?p=4174) (100%)
* [Как просмотреть содержимое ключей и сертификатов SSL](https://zalinux.ru/?p=4178) (77.7%)
* [Лучшие терминальные мультиплексные инструменты](https://zalinux.ru/?p=76) (50%)
* [Как просмотреть или отредактировать метаданные pdf или изображениях из командной строки Linux](https://zalinux.ru/?p=115) (50%)
* [Как установить 7zip на Linux](https://zalinux.ru/?p=224) (50%)
* [Как передавать данные между контейнером Docker и хостом](https://zalinux.ru/?p=1046) (RANDOM - 50%)